A honeypot-driven cyber incident monitor: lessons learned and steps ahead

Emmanouil Vasilomanolakis[†*], Shankar Karuppayah[†§], Panayotis Kikiras[*], Max Mühlhäuser[†]

[†]Telecooperation Group, TU Darmstadt - CASED first.last@cased.de

[§]National Advanced IPv6 Center, Universiti Sains Malaysia shankar@nav6.usm.my

[*]AGT International, Darmstadt, Germany pkikiras@agtinternational.com

ABSTRACT

In recent years, the amount and the sophistication of cyber attacks has increased significantly. This creates a plethora of challenges from a security perspective. First, for the efficient monitoring of a network, the generated alerts need to be presented and summarized in a meaningful manner. Second, additional analytics are required to identify sophisticated and correlated attacks. In particular, the detection of correlated attacks requires collaboration between different monitoring points. Cyber incident monitors are platforms utilized for supporting the tasks of network administrators and provide an initial step towards coping with the aforementioned challenges.

In this paper, we present our cyber incident monitor *TraCINg*. *TraCINg* obtains alert data from honeypot sensors distributed across all over the world. The main contribution of this paper is a thoughtful discussion of the lessons learned, both from a design rational perspective as well as from the analysis of data gathered during a five month deployment period. Furthermore, we show that even with a relatively small number of deployed sensors, it is possible to detect correlated attacks that target multiple sensors.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; C.2.0 [**Computer-Communication Networks**]: General: Security and Protection

General Terms

Security

Keywords

Incident Monitor; Honeypot; Alert Correlation; Cyber Security

1. INTRODUCTION

With the increase of cyber attacks in both numbers, as well as in sophistication, it is becoming evident that more advanced techniques and mechanisms are required to detect attacks and to identify new trends and patterns in the adversaries' strategies. This also implies that relying only upon traditional lines of defense, such as Intrusion Detection Systems (IDSs) and dynamic firewalls alone, would not be able to provide a holistic coverage on detecting novel and emerging patterns of attacks [20].

Honeypots are systems whose value lies solely in being probed, attacked or compromised [16]. They can complement other detection mechanisms, e.g., IDSs, by providing a more active and in-depth view on attackers' activities. At the same time, by definition, honeypots produce zero false positives and they are able to detect even unknown and novel attacks. They can be classified based on the level of interaction that is offered to attackers. On the one hand, *high-interaction* honeypots are real systems that exhibit certain vulnerabilities. Thus, they require constant monitoring and their maintenance is very demanding. On the other hand, *low-interaction* honeypots, simulate network operations on the TCP/IP stack level, and therefore can provide a lightweight and straightforward defense mechanism.

Feeding honeypot alert data into a cyber-incident monitor provides a number of significant benefits. As mentioned above, in contrast to IDSs, honeypots do not produce false positives. Therefore, the security administrator can focus on real attacks rather than speculating from the observed results. In addition, observing epidemic behavior, e.g., malware spreading, becomes more likely when analyzing attacks from several collaborating honeypots.

In essence, the inclusion of different type of alerts, e.g., from IDSs and honeypots, increases the affluence of new data to be processed by the administrators. However, due to the different focus and content of alerts generated by different systems, integrating these alerts into a single platform, creates a multitude of challenges. Cyber incident monitoring systems are developed to overcome this problem by providing an aggregation and visualization interface to unify the various alert generators into a single system that assists the user to make informed decisions. Starting from a broad overview, users can decide to dive into specific alerts to assess a particular reported cyber incident.

Not much work has been done that specifically focuses on the design of a cyber incident monitor, or the lessons learned from its usage. Dshield[1], and honeymap[2] are two examples of monitors that publish their data openly through the Internet. In [7], Katti et al. provide a wide-scale analysis focusing on correlated attacks. Even though their work can be considered slightly outdated, it is the basis on which we establish the definitions of correlated attacks (cf. Section 3.3). Some work has also been conducted towards providing an analysis of the lessons learned from such distributed monitoring deployments [9, 11] and honeypots [8]. However, most of this work focuses only on presenting basic attack statistics and does not consider correlated attacks.

In this paper, we present *TraCINg*[3], which stands for TU Darmstadt Cyber Incident moNitor, and is an open-source centralized cyber incident monitor that supports a number of different types of honeypots. *TraCINg* is able to visualize attacks, provide statistics, present the geo-location information of malicious users, and perform several other user-centric operations. We explain the design rational of our system and discuss the initial results from a five month deployment period. Our results, indicate a multitude of interesting findings. First, we notice many attack trends such as the most popular protocols and ports, the most persistent countries of origin, etc. Second, we observe that even with a relatively small deployment of sensors, it is still possible to detect attacks manifested by the same source on multiple sensors. Furthermore, these correlated attacks motivate the need for collaboration between different monitoring points in detecting targeted or distributed attacks.

The remainder of this paper is organized as follows: In Section 2, we describe our system's design and provide justification behind the various design choices that we made. Section 3, provides insights and an analysis of our findings for a deployment period of five months. Section 4, discusses the lessons learned from our experience with the cyber incident monitor, problems that we faced, as well as hints on how to overcome them. Finally, Section 5 concludes this paper.

2. SYSTEM DESIGN: TRACING

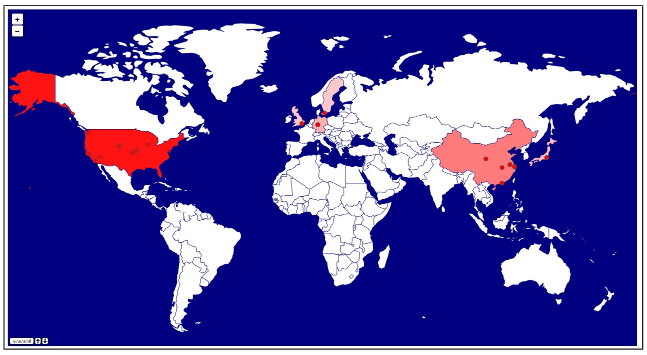
In this section, we provide a description of *TraCINg*'s design, including three main parts: the *architecture*, the different types of supported honeypot *sensors*, and the *alert* output.

2.1 Architecture and Design

TraCINg is an open-source centralized cyber incident monitor that obtains alert data from geographically distributed honeypot sensors. In our system, sensors are currently deployed in three different continents, namely: Europe, Asia and the America. *TraCINg* follows a classic client-server architecture and uses an *HTTPS* server for receiving data, along with a Public Key Infrastructure (PKI) for the authentication of the sensors. The front-end, written in *node.js*[17], supports the integration of, e.g., OpenStreetMap [6] for the visualization of attack origins and targets (cf. Figure 1(b)) and VirusTotal[4] for obtaining additional information on identified malware.

[1]http://dshield.org
[2]http://map.honeynet.org
[3]ssi.cased.de
[4]virustotal.com

All gathered information is available to security administrators via a user-centric GUI with a plethora of different views (two of them shown in Figure 1) and functions, e.g., the creation of statistics, on-demand detailed information, live and replay mode of the alert data, etc.


(a) Default TraCINg view

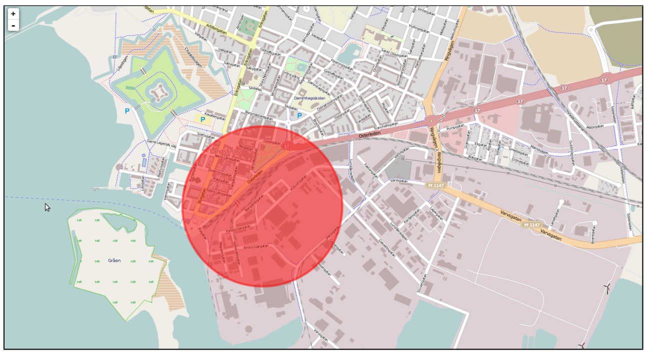

(b) OpenStreetMap integration

Figure 1: TraCINg GUI examples

2.2 Sensors

TraCINg supports and utilizes a number of sensors that can be distinguished into two main types: *dionaea* honeypots and mobile honeypots. Here, we shortly discuss these types of honeypots along with the implementation specifics of our sensors.

Dionaea Honeypots.

Our main detection mechanism uses the *dionaea*[5] low-interaction honeypot, that is considered to be the state of the art in its honeypot class, i.e., low-interaction [15]. We utilized dionaea honeypots in two different ways. First, we deployed dionaea on regular machines, i.e., virtual machines (VMs), as well as Raspberry Pis. The latter allows for an easy copying of systems, to be able to reuse them, and introduces convenient plug-n-play support for the sensors.

Second, we also deployed some dionaea sensors as cloud instances. This is important for being able to monitor attack traffic that may be specific to geographical regions. Thus, via the utilization of cloud providers, we were able to deploy our sensors in different continents. Furthermore, cloud services provide resilience and uptime reliability for our sensors which ensures uninterrupted monitoring.

[5]http://dionaea.carnivore.it

Mobile Honeypots.

TraCINg also provides support for obtaining data from mobile honeypots, i.e., from *HosTaGe* which is developed by us in our former work [19, 18]. *HosTaGe* is a lightweight low-interaction honeypot for mobile devices for detecting malicious wireless network environments. The motivational idea behind *HosTaGe* is to alert users on the security status of the wireless network they are connected to. In addition, the combination of *HosTaGe*'s collaboration techniques and synchronization with *TraCINg*, creates additional benefits for the users. For instance, users can learn from others, about the global security status of wireless networks in their city, or country. Moreover, the diverse data that is sent to *TraCINg* can also provide us with valuable and more accurate input for the detection of correlated attacks and latest trends from the behavior of malicious users and malware.

2.3 Alerts

TraCINg supports input from any type of honeypot or IDS, with the only restriction being to utilize the respective input interface for submitted alerts. Sensors need to convert their alerts into a well-determined JSON[6] format that is being supported by the input interface. Therefore, it is straightforward to add support for other honeypots. For instance, to enable *TraCINg* support for the dionaea honeypot, we implemented the respective alert export functionality as an internal dionaea module.

Another important property, that is related to alerts, is the frequency of the exchanged alerts. In the current version of our system, alert data is sent instantaneously upon the detection of an attack by the honeypot to *TraCINg*, except for *HosTaGe* honeypots in which, the exchange frequency is determined by the user or by the auto-upload functionality (when the mobile device has Internet connectivity, e.g., via a wireless network). Take note that the frequency of the exchanged alerts could also lead to some attacks on the sensors (cf. Section 4 — *Probe-Response attacks*).

The generated alerts that are submitted to *TraCINg* contain the following attributes:

- *Time-stamp*: The date and time specifics of an attack.
- *Id*: A unique identifier for each alert.
- *Sensor Type*: A field that differentiates between different honeypots, e.g., dionaea, HosTaGe, etc.
- *IP*: The source and destination IP address of attacks (this information is excluded from the end-users' perspective, i.e., from the GUI, to maintain privacy).
- *Ports*: The source and destination port of attacks.
- *Attack type*: The type of the detected attack, e.g., a portscan, a shellcode injection, etc.
- *Geo-location Information*: The geo-IP information, and the name of the city and country.
- *Authorization Status*: A boolean id that indicates whether a sensor posses a signed certificate from the main TraCINg Certificate Authority (CA) via the usage of a PKI.
- *MD5 hash*: The MD5 hash of the malware collected from a honeypot (when applicable).
- *Log*: Whenever possible, the whole communication between the attacker and the sensor is logged, and can be presented to the user on demand.

[6]http://www.json.org

3. ALERT DATA ANALYSIS

In this section, we present the results of a five month observation period (March to July 2014) of *TraCINg* with a total of five sensors deployed in three different continents.

3.1 System Setup

During the period of our analysis, we collected data from $five$ different honeypots that were continuously monitoring and sending alerts to *TraCINg*. The specifics of our deployed sensors are summarized in Table 1. In more details, two sensors were located in Malaysia within a /24 sub-network. In addition, two sensors were deployed as cloud instances in USA within different /8 networks. Finally, one sensor was deployed in Greece. As we will also discuss in Section 3.3, the dynamics of the alert data are important for the subsequent analysis of the attacks, especially in a correlated manner.

Sensor Name	Country	Common Subnet Prefix
MY-01	Malaysia	/24
MY-02	Malaysia	
USA-01	USA	/8
USA-02	USA	
GR	Greece	-

Table 1: Sensor description

For our analysis, we restrict our evaluation to the stationary deployed dionaea honeypots only. Data gathered from our mobile honeypots was removed as the number of users making use of the mobile honeypot, at that time, was immature (and restricted only to Germany). Thus this would introduce bias into our analysis. Nevertheless, our analysis shows that even with a relatively low number of sensors, a large amount of distributed and correlated attacks can be detected.

A preliminary overview of recent results gathered from usage of mobile honeypots, between March and May of 2015, is as following. Approximately 1000 alerts were sent from four different countries (i.e., Germany, United Arab Emirates, Italy, and Columbia). In a glance, 62% of the attacks were shellcode injection, 24% were targeting the TELNET protocol, 6.6% were targeting HTTP, and finally 4% were MySQL brute-force attacks.

3.2 Data analysis

In total, our sensors recorded 898,570 alerts during the examined period, from 30,101 distinct IP addresses, having their origin in 146 different countries. However, it is interesting to note that attacks from USA and China alone, represented about 80% of the total number of alerts recorded by our system. Nevertheless, we have verified that this observation is not influenced by the geographical location of the deployed sensors.

Table 2, presents a summary of the most popular attack types along with the number of attack occurrences. The attack description column of the table depicts the specifics of the attacks. In this case *MySQL*, *MS SQL*, *VOIP*, and *SMB* refer to malicious activity specific to these protocols,

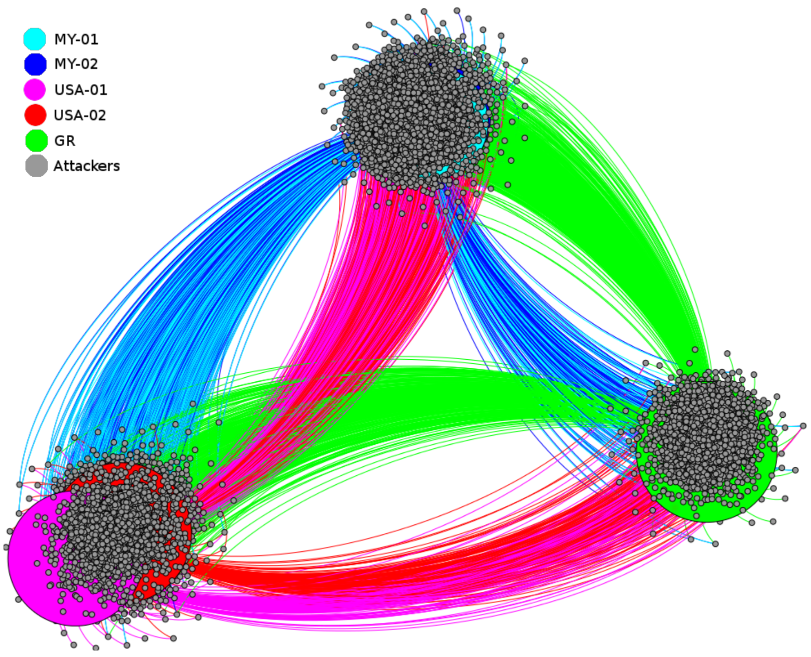


Figure 2: Graph representation of the alert data: attackers clustered close to their main targets and single-dimensional correlation seen as edges connecting to neighbor clusters

Attack Description	Total Number of Attacks
Portscan	507,571
MySQL	156,960
Transport Layer	116,414
VoIP	84,641
SMB	30,239
MS SQL	2,325
Shellcode Injection	420

Table 2: Most popular attack types and the corresponding number of occurrences

Port	Protocol/Service	Number of Attacks
135	RPC	24,667
139	NetBIOS	20,249
23	Telnet	11,058
80	HTTP	10,735
445	SMB	9,294
443	HTTPS	3,400
25	SMTP	2,558
21	FTP	1,658
110	POP3	1,153
143	IMAP	597

Table 3: Top 10 attacked ports and protocols

e.g., a brute-force attack. Moreover, the *Transport Layer* describes cases in which the adversary connected to a port, but no further activity was possible due to honeypot-related limitations, e.g., dionaea not being able to handle the connection. With respect to dionaea's *VOIP* emulation capabilities [21], the majority of the detections were brute-force attacks and scans conducted with tools such as *SipCli*[7].

In addition, Table 3 shows the most targeted ports and protocols. A number of findings come as a result of our analysis. First, around 56% of the total attacks were portscans. This can be considered as something that is expected, especially with the rise of open source Internet-wide network scanners, e.g., ZMap [4]. In addition, it should be noted that bias can be introduced from such tools when utilized by researchers. Nevertheless, we consider this insignificant with regards to the overall analysis. Second, most of the detected MySQL attacks were brute-force attacks using default user-names and passwords. The most prominent one being a combination of *root* as a user-name along with a blank password. Moreover, Table 3 indicates that several attacks targeting Windows OS specific protocols and services, e.g., the *MS-RPC*, and *NetBIOS*, are still prominent. This also confirms that several old worm variants, e.g., Conficker [12], still hold an imposing position in the overall attack propagation scene. Lastly, we identified that a number of the attacks that were targeting *Telnet* were conducted by insecure/infected embedded devices, e.g., IP web-cams.

3.3 Correlation of attacks

We classify correlated attacks in a twofold manner, differentiating between *single-dimensional* and a *two-dimensional* correlation of attacks, by following a *similarity*-based strategy [5]. *Single-dimensional* correlation, groups attacks with the same origin, e.g., the same source IP address. Thus, it can be defined as the set of alerts originating from the same source IP address that target more than one sensor throughout our observation, i.e., the entire observed duration of five months. *Two-dimensional* correlation includes time as an additional parameter, i.e., attacks that are observed within a specific time window.

[7]SipCli VoIP audit tool: http://www.kaplansoft.com/SipCli

Single-dimensional correlation.

The analyzed dataset can be modeled as a directed graph $G = (V, E)$, where the set of nodes V represent all IP addresses involved in the detection process, i.e., both sensors and malicious users. The origin and the target nodes of an attack, are represented as a set of directed edges (u, v) with $u, v \in V$, that exist between the nodes.

Figure 2, is a representation of our alert dataset, that depicts two major findings: attack origins can be clustered, by vicinity, into three clusters and the single-dimensional correlation can be seen as the edges that connect to neighboring clusters. Specifically, the dataset was transformed to a directed graph with 30, 106 nodes representing all distinct IP addresses (both sensors and malicious users), while edges correspond to the respective connections, i.e., adversaries connecting to the sensors. Figure 2 depicts, in a glance, an overview of the activities in our dataset.

The five different sensors (differentiated by distinct colors) converge into the three main clusters. The clustering is done based on the geo-location information of the sensors (cf. Section 3.1). As such, from the $five$ deployed sensors, four of them were coupled into clusters of two. This is also explained in our system setup description in Section 3.1. The Malaysian sensors (within a /24 network) create tightly coupled clusters due to the high percentage of common attackers. In addition, we observe that the two sensors in the USA, even though having distinct /8 networks, are also close to each other. Figure 2, also clusters the attackers based on the intensity of the alerts/attacks observed originating from them, i.e., nodes are placed closer to the sensors they attacked intensively. Moreover, *single-dimensional* correlation is observed when edges of a cluster (same color), are connecting to neighboring clustering groups, i.e., an adversary targeting multiple sensors.

A similar analysis, from another perspective, on the ratio of *single-dimensional* correlation is shown in Figure 3. In more details, we plot the relationship between the percentage of unique attackers and the targeted sensors. Almost 50% of the total number of attacks target at least two different sensors, during the $five$ month period under investigation. However, a portion of this finding might be due to the 'closeness' of two sensors, i.e., sensors located within the same /24 sub-network. Nevertheless, as we discuss in the following section, even with the inclusion of time as a parameter, we observe an almost continuous attacking behavior on multiple sensors simultaneously.

Two-dimensional correlation.

In *single-dimensional* correlation, the basis for our analysis is exclusively based on the source IP address of the adversaries. While this is reasonable, we argue that the time frame in which attacks take place is also an important factor that needs to be taken into account. There is a significant difference between distributed attacks targeting multiple widespread sensors within a small time window and the ones that target multiple sensors across longer time period, i.e., days or even weeks.

Figure 4 presents the number of unique attackers targeting multiple sensors within a short time frame. In more details, this two-dimensional correlation, is measured with a sliding window of *one* hour (measurements taken every 30 minutes). The 30 minutes time interval takes into account the state of the art in IPv4 network scanning. As of now

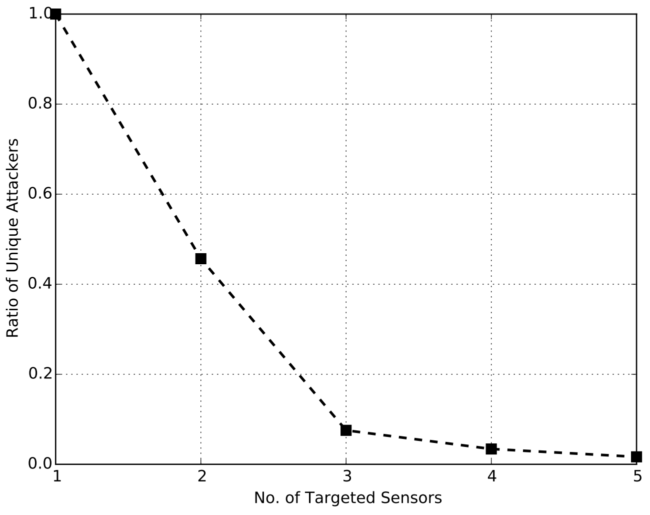


Figure 3: Ratio of unique attackers targeting multiple sensors

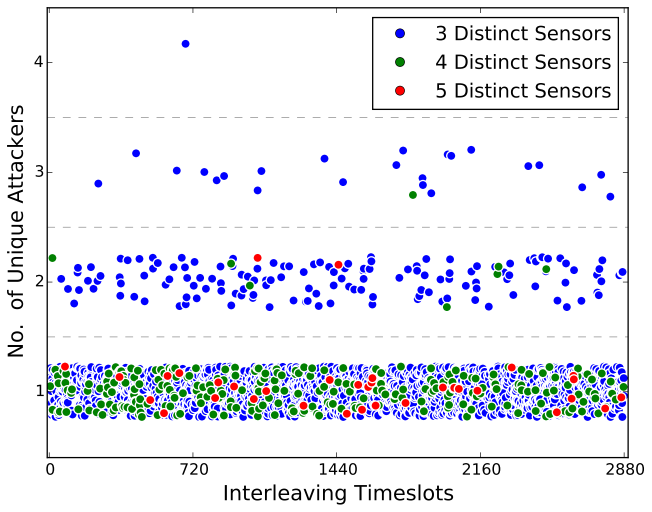


Figure 4: Unique attackers within a sliding window of one hour and with measurements taken every 30 minutes

one of the fastest Internet-wide scanners is a able to scan the IPv4 address space in approximately 45 minutes (via a random cyclic selection of IP addresses) [4]. Thus, with our chosen time interval we attempt to reduce bias introduced by researchers' activities utilizing such mechanisms.

We note that for the sake of clarity, we applied vertical jitter to the points in the plot. Moreover, we also restricted the figure to only two months (May to July 2014) of the overall period. Nevertheless, similar activity was observed for the removed time-frame. Throughout the observations, we notice a pattern of attacking behavior in which, at least *one* unique attacker is targeting *three* or $four$ different sensors (within a 30 minutes observation window). Furthermore, there are also cases in which, up to $four$ different adversaries targeted *three* sensors. Cases like this, might indicate the discovery of a new vulnerability and thus the possibility of extensive scans over the Internet for exploitable insecure systems. Finally, the majority of two-dimensional correlation was portscans. In addition, for the cases of *three* and *five* overlapping sensors, we also detected *MySQL* and *VOIP* brute-force attacks.

4. LESSONS LEARNED & FUTURE WORK

In this section, we discuss the lessons learned and the shortcomings that we encountered during the design, as well as the deployment phase of our cyber incident monitor.

Correlated attacks and number of sensors.

Although our system is already able to detect correlated attacks, we believe that its accuracy can be further improved. When sensors are sparsely deployed in the IP space, we may miss out correlated attacks simply because the time taken to observe such an attack is too long, e.g., more than 30 minutes. Therefore, the accuracy and efficiency of our system, i.e., the required time to detect correlated attacks, can be improved by increasing the number of sensors. This ensures that epidemic worm propagation and new vulnerabilities can be detected at a faster rate than the current implementation. Thus, we plan to deploy more sensors and distribute them uniformly across the various classes of IP address, i.e., /8 and /16 sub-networks. Moreover, the idea behind collecting alerts from mobile honeypots (cf. Section 2.2) also further bridges this gap of being able to detect correlated attacks more accurately and in a shorter time frame.

Performance.

While performing stress tests to our system with a high load of incoming traffic, e.g., more than $50K$ of alerts per day, we observed a bottleneck that relied in the utilized database (which was not always able to handle these requests). Thus, our initial decision to use a *SQLite* database during the proof of concept phase was no longer relevant with an increased number of alerts or sensors. We migrated our alerts and re-deployed on *MongoDB* that is now able to cater our current scenarios better and also facilitate future expansions.

An interesting real world example of such an attack peak was during the first quarter of April 2014. During that timespan our cyber incident monitor had been receiving almost $40,000$ alerts per day, with alerts being either portscans or targeting the transport layer. We assume that this was not a coincidence but rather related with the well known *heartbleed* OpenSSL vulnerability [2], that was discovered during the aforementioned days.

Scalability.

Although our system design provides real-time visualization of the alerts, which is contributed by its centralized architecture, it also presents itself as a single point of failure. Moreover, the architecture itself will turn into a bottleneck when the number of new sensors is massively increased, hence affecting the scalability of the system itself. We envision a new version of our system that utilizes a completely decentralized architecture and will be able to include additional new sensors while providing increased resilience, redundancy and scalability to the overall system.

Probe-Response attacks.

Recently research has been conducted [13, 14, 3] that focuses on methods that can be utilized by attackers to successfully detect monitoring sensors that publish their findings publicly via the Internet. In these attacks, so-called *probe-response* attacks, the adversary produces alerts that contain a unique watermark and starts disseminating them over the IP address space that the attacker considers as possible for including sensors. The malicious user can subsequently compare the results produced from the cyber incident monitor, for the specified time period in which the probe-response attack is performed, and thus be able to create a list of possible addresses of sensors via a *reductio ad absurdum* method.

Probe-response attacks have not been implemented in the wild by malware. However, a malware with such capabilities would be able to evade a large amount of cyber-incident sensors and thus it will be more likely to spread unnoticed for a longer time period. Moreover, as reported in [14], defending against such attacks is not a trivial task. As such, handling probe-response attacks will be one of the main considerations of our future work. For instance, countermeasures might include the addition of *noise data* to the publicly published results, by following a uniform random distribution. In addition, these attacks assume a public exposure of the alert data, and thus can be avoided via an internal usage of *TraCINg*.

Honeypot-specific issues.

During our honeypot deployment we encountered various problems that are related to the dionaea honeypot itself, and should be considered for such deployments. First, nowadays many honeypots (including dionaea, nepenthes [1], etc.) can be detected relatively easily with network scanning tools such as nmap [10]. Therefore, researchers should consider changing the default parameters of such honeypots. For instance, a honeypot with many conflicting ports active (e.g., Windows and Linux OS specific ports simultaneously active), is easier to be detected. Second, in some cases, portscans have to be handled carefully. In more details, a default dionaea installation will react to a portscan by activating services for each of the scanned ports. In the worst case scenario, this may result to approximately 65,000 services (i.e., all the possible port numbers) activated for each scanned IP address. Thus, techniques to prevent such cases should be taken into account (e.g., dionaea does offer solutions for this problem).

Privacy and Trust.

As part of our future goal to enrich our system with a more diverse selection of sensors, we need to ensure that the privacy of data from external contributors, e.g., individual users or organizations, is assured. We plan to provide this by including a level of filtering and obfuscation that ensures that private information, e.g., real IP addresses, are filtered and not stored in our system's database. Inclusion of new and unknown sensors to our system may lead to internal attacks, e.g., alert spoofing. These attacks could in turn affect the accuracy of the entire system. Nevertheless, we plan to overcome such attacks by deploying a trust mechanism between the participating sensors along with the existing PKI, to ensure that the authenticity of sensors is always ensured before accepting alerts in our system.

Distribution of Sensors.

An observation made after deploying our sensors in the cloud was that in some cases, we received large amounts of alerts due to attacks that seemed to be targeting particularly the cloud vendor itself. More specifically, we detected a Distributed Denial of Service (DDoS) while we were experimenting with the deployment of sensors in the same cloud provider; yet in totally different geographical locations. Therefore, it should be noted that although many cloud providers offer deployment of instances all over the world, the range of IP addresses that is utilized is specific and in many cases, well known to malicious entities. Moreover, if sensors are only placed within a single cloud provider, this may introduce bias in the final results; attacks that targeted only the cloud provider(s) might be misinterpreted as though they were attacks taking place all over the globe.

5. CONCLUSION

Sophisticated adversaries as well as the increased number of attacks in general, create challenges both in the detection and the analysis phase. Furthermore, the successful detection of correlated attacks, requires the collaboration between different monitoring points. Cyber incident monitors provide a way for researchers and security administrators to efficiently create a holistic view of their monitored networks.

In this paper, we presented a cyber incident monitor, *TraCINg*, that is driven by honeypots. We discuss about its architecture and the various design choices that were made. In addition, we provide an analysis of data gathered from *TraCINg*, during a period of $five$ months. Our alert data analysis indicates a plethora of findings: first, many attacks target protocols that are considered as deprecated, e.g., Telnet, are still prominent, while newer protocols, e.g., VOIP, also begin to experience increased attacks. Second, we observe large-scale portscans that, as a result of our correlation analysis, appear to target a wide range of IP addresses.

6. ACKNOWLEDGMENTS

We would like to thank *ISLAB* from the National Center for Scientific Research "Demokritos" (Greece), as well as the Malaysian Research and Education Network (MYREN) for their support during the honeypot deployment phase.

7. REFERENCES

[1] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The nepenthes platform: An efficient approach to collect malware. In *Recent Advances in Intrusion Detection*, pages 165–184. Springer, 2006.

[2] M. Carvalho, J. Demott, R. Ford, and W. David. Heartbleed 101. *Security & Privacy, IEEE*, 12(4):63–67, 2014.

[3] S. Cheung. Securing collaborative intrusion detection systems. *IEEE Security and Privacy*, 9(6):36–42, 2011.

[4] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *Proceedings of the 22nd USENIX Security Symposium*, pages 605–619, 2013.

[5] H. T. Elshoush and I. M. Osman. Alert correlation in collaborative intelligent intrusion detection systems—A survey. *Applied Soft Computing*, 11(7):4349–4365, Oct. 2011.

[6] M. Haklay and P. Weber. OpenStreetMap: User-Generated Street Maps. Pervasive Computing. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[7] S. Katti, B. Krishnamurthy, and D. Katabi. Collaborating against common enemies. In *5th ACM SIGCOMM Conference on Internet Measurement - IMC*, New York, New York, USA, 2005. ACM Press.

[8] G. Kelly and D. Gan. Analysis of Attacks Using a Honeypot. In *International Cybercrime, Security and Digital Forensics Conference*. Springer, 2011.

[9] I. Koniaris, G. Papadimitriou, P. Nicopolitidis, and M. Obaidat. Honeypots deployment for the analysis and visualization of malware activity and malicious connections. In *International Conference on Communications (ICC)*, pages 1819–1824. IEEE, 2014.

[10] G. F. Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure, 2009.

[11] K. Ohno, H. Ko, and K. Koizumi. IPMatrix : An Effective Visualization Framework for Cyber Threat Monitoring. In *Proceedings of the Ninth International Conference on Information Visualisation*, 2005.

[12] S. Shin and G. Gu. Conficker and beyond: a large-scale empirical study. In *26th Annual Computer Security Applications Conference*, pages 151–160, 2010.

[13] Y. Shinoda, K. Ikai, and M. Itoh. Vulnerabilities of passive internet threat monitors. In *14th USENIX Security Symposium*, pages 209–224, 2005.

[14] V. Shmatikov and M.-H. Wang. Security against probe-response attacks in collaborative intrusion detection. In *Workshop on Large scale attack defense - LSAD*, pages 129–136, New York, New York, USA, 2007. ACM.

[15] T. Sochor and M. Zuzcak. Study of internet threats and attack methods using honeypots and honeynets. In *Computer Networks*, pages 118–127. Springer, 2014.

[16] L. Spitzner. Honeypots : Catching the Insider Threat. In *Computer Security Applications Conference*, pages 170–179. IEEE, 2003.

[17] S. Tilkov and S. Vinoski. Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):0080–83, 2010.

[18] E. Vasilomanolakis, S. Karuppayah, M. Fischer, M. Mühlhäuser, M. Plasoianu, L. Pandikow, and W. Pfeiffer. This Network is Infected : HosTaGe - a Low-Interaction Honeypot for Mobile Devices. In *Security and privacy in smartphones & mobile devices*, pages 43–48. ACM, 2013.

[19] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer. HosTaGe: a Mobile Honeypot for Collaborative Defense. In *7th International Conference on Security of Information and Networks (SIN)*. ACM, 2014.

[20] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer. Taxonomy and Survey of Collaborative Intrusion Detection. *ACM Computing Surveys*, 47(4):33, 2015.

[21] M. Voznak, J. Safarik, and F. Rezac. Threat Prevention and Intrusion Detection in VoIP Infrastructures. *International Journal of Mathematics and Computers in Simulation*, 7(1), 2013.